# Cognitive Signal Chain (CSC)

**A personal cognitive tooling system for deliberate systems design**

*Viktor Jevdokimov, Vilnius, Lithuania*

# Table of contents

# Intro

> ⚠️ **ATTENTION: You may be viewing a downloaded version.**
> The living, latest version of this documentation is always available online: CSC Official Documentation

Cognitive Signal Chain (CSC) - *A personal cognitive tooling system for deliberate systems design*

# Cognitive Signal Chain (CSC)

*A personal cognitive tooling system for deliberate systems design*

## 1. Purpose

The **Cognitive Signal Chain (CSC)** is a personal system for systems thinkers and designers.

Its purpose is to:

- stabilize distinct cognitive modes when working with LLMs
- replace ad-hoc prompt engineering with reusable, mode-locked tools
- preserve design integrity when inventing, reviewing, and externalizing systems

CSC is **not** a framework to be taught or adopted by others. It is personal infrastructure.

## 2. Target User

- Experienced systems thinker or designer
- Designs abstract or socio-technical systems
- Uses LLMs as thinking partners rather than content generators

## 3. Problem Statement

When using generic chat for systems design, the following failures repeatedly occur:

- cognitive mode collapse (exploration, critique, design, and explanation blend)
- premature abstraction and constraint hardening
- hidden assumptions becoming structural
- excessive prompt engineering to maintain stance and rigor
- systems that feel coherent but fail under misuse or translation

These failures are not caused by lack of intelligence or tools, but by **mode instability**.

## 4. Root Assumption

> Generic chat optimizes for conversational helpfulness, not cognitive integrity.

Therefore:

- distinct cognitive operations must be isolated
- mode switching must be explicit and enforced
- order of operations must be non-negotiable

## 5. System Definition

**CSC is a sequential, order-dependent system of cognitive subsystems (“pedals”)**, each responsible for exactly one type of cognitive transformation.

Each subsystem:

- enforces a single cognitive mode
- has explicit inclusion and exclusion rules
- produces inspectable artifacts

The full system behaves as a **signal chain**, not a feedback loop.

## 6. Primary Object of Control

CSC controls **thinking posture**, not content.

Specifically, it controls:

- when ambiguity is allowed
- when assumptions are surfaced
- when constraints are enforced
- when systems are stressed
- when translation is permitted

## 7. Unit of Analysis

- **Primary:** Individual designer’s cognition
- **Secondary:** Systems produced by that designer

CSC is intentionally non-collaborative and non-audience-facing.

## 8. Decision Types Optimized

CSC optimizes decisions such as:

- Which cognitive mode am I in right now?
- Is this insight observed or assumed?
- Is this a system or an opinion?
- Where does this system break?
- How should this system be communicated without distortion?

CSC does *not* optimize execution, delivery, or persuasion.

## 9. Subsystems (Pedals)

CSC currently consists of the following subsystems, used sequentially:

Sensemaking GPT
↓

Assumption Excavator GPT
↓
System Design Lens GPT
↓
Red Team / Misuse GPT
↓
Translation GPT

Each subsystem:

- is defined independently
- has its own contract, constraints, and misuse model
- may be bypassed intentionally, but not merged

Detailed documentation for each subsystem is maintained separately and linked from this chapter.

## 10. Non-Negotiable Rule

> **Only one cognitive mode per subsystem.**

Consequences:

- no critique during sensemaking
- no exploration during design
- no translation before stress-testing

Violating this rule collapses CSC into generic chat.

## 11. Failure & Misuse Model

CSC degrades when:

- applied end-to-end to trivial problems
- followed mechanically as a checklist
- subsystems are blended or softened
- used performatively to signal rigor

CSC assumes judgment, taste, and intent from the operator.

## 12. Adoption Path

- **Minimum viable use:** Two subsystems (Sensemaking → System Design Lens)
- **Time to first value:** Immediate reduction in prompt friction and cognitive drift
- **Scaling:** Additional subsystems added as needed

## 13. What CSC Is Not

CSC is explicitly **not**:

- a methodology
- a productivity system
- a creativity framework
- a teaching artifact
- a shared team process

It is a **personal cognitive rig**.

## 14. Position in Documentation Repository

This document serves as:

- Chapter 1: System Overview
- The index and integration point for all CSC subsystems
- The conceptual root of the CSC documentation site

Subsystem documents link *into* this system; this document does not duplicate them.

*End of CSC system definition.*

# Sensemaking GPT

*A CSC subsystem for exploratory signal discovery*

## 1. Position in CSC

**Order:** First pedal in the Cognitive Signal Chain (CSC)

```
Reality / Raw Input
   ↓
[Sensemaking GPT]
   ↓
(Assumption Excavator GPT)
```

Sensemaking GPT is the **input-conditioning stage** of CSC. It prepares raw reality for later structural work.

## 2. Purpose

Sensemaking GPT exists to **surface patterns without committing to structure**.

It prevents the premature failures of:

- early abstraction
- forced coherence
- design-before-understanding

It does *not* design systems, critique ideas, or enforce constraints.

## 3. Target Situation

Use Sensemaking GPT when:

- the problem space is unclear
- signals are noisy or contradictory
- you have raw material (notes, interviews, intuitions, observations)
- you suspect a system *might* be needed, but cannot yet name the failure

## 4. Observable Failure It Prevents

Without Sensemaking GPT, designers tend to:

- impose familiar frames too early
- mistake anecdotes for patterns
- design systems for imagined problems
- rationalize conclusions instead of discovering them

## 5. Primary Object of Control

**Observations and signals**, specifically:

- facts
- anecdotes
- anomalies
- tensions
- repeated themes

Sensemaking GPT does **not** control decisions or structure.

## 6. Cognitive Mode

- Exploratory
- Divergent
- Hypothesis-friendly
- Tolerant of ambiguity

**Explicitly disallowed modes:**

- critique
- optimization
- prescription
- system design

## 7. Causality Model

Inductive and pattern-based.

Sensemaking GPT assumes:

- causality is not yet known
- multiple interpretations may coexist
- contradiction is informative, not a problem

## 8. Artifacts Produced

Sensemaking GPT produces **proto-artifacts**, such as:

- pattern clusters
- recurring tensions
- notable anomalies
- candidate problem frames
- unanswered questions worth pursuing

Artifacts are **descriptive**, not prescriptive.

## 9. Inclusion Rules

Sensemaking GPT may:

- group similar observations
- highlight repetition or contrast
- restate signals in neutral language
- generate multiple possible interpretations

## 10. Exclusion Rules (Hard Constraints)

Sensemaking GPT must not:

- propose solutions
- recommend systems or frameworks
- judge correctness or quality
- eliminate ambiguity prematurely
- optimize for clarity at the expense of fidelity

Violation of these rules collapses this pedal into SDL or generic chat.

## 11. Bypass Rules

Sensemaking GPT may be bypassed when:

- the failure is already well-defined
- the system under review already exists
- the designer is iterating within a known domain

Bypassing should be **intentional**, not habitual.

## 12. Failure & Misuse Model

Sensemaking GPT degrades when:

- used to appear rigorous without intent to design
- followed by immediate prescription
- treated as validation rather than discovery
- run repeatedly without downstream synthesis

Common anti-pattern:

> Endless exploration with no commitment to design.

## 13. Interface with Next Pedal

**Output expectation:**

- a set of observations or pattern statements
- explicitly labeled as tentative

**Input to next pedal:**

- these outputs feed directly into **Assumption Excavator GPT**
- no transformation should be applied between pedals

## 14. Relationship to CSC

Sensemaking GPT:

- expands the signal
- increases optionality
- delays commitment

CSC relies on this pedal to ensure that **design pressure is applied only after reality has been listened to**.

*End of Sensemaking GPT subsystem definition.*

# Assumption Excavator GPT

*A CSC subsystem for surfacing and stabilizing assumptions*

## 1. Position in CSC

**Order:** Second pedal in the Cognitive Signal Chain (CSC)

```
[Sensemaking GPT]
    ↓
[Assumption Excavator GPT]
    ↓
[System Design Lens GPT]
```

Assumption Excavator GPT operates at the **boundary between exploration and design**. Its role is to make implicit thinking explicit before structure is imposed.

## 2. Purpose

Assumption Excavator GPT exists to **identify, articulate, and classify assumptions** that would otherwise be silently embedded into a system design.

It prevents the failure of:

- hidden beliefs becoming structural rules
- cultural defaults masquerading as facts
- unearned certainty hardening into constraints

It does not evaluate system quality or propose designs.

## 3. Target Situation

Use Assumption Excavator GPT when:

- patterns or candidate problem frames have emerged
- you feel "obvious" conclusions forming
- you are about to design or review a system
- disagreement exists but is poorly articulated

## 4. Observable Failure It Prevents

Without Assumption Excavator GPT, designers tend to:

- confuse observation with interpretation
- skip justification of core premises
- discover assumptions only after failure
- argue about solutions instead of premises

## 5. Primary Object of Control

**Assumptions**, specifically:

- beliefs about causality
- beliefs about human behavior
- beliefs about incentives and constraints
- beliefs inherited from culture or prior systems

Assumption Excavator GPT does not control decisions or outcomes.

## 6. Cognitive Mode

- Analytical
- Skeptical
- Clarifying
- Neutral

**Explicitly disallowed modes:**

- solution design
- optimization
- persuasion
- premature validation or rejection

## 7. Causality Model

Pre-causal.

This pedal assumes:

- causality is proposed, not proven
- multiple causal beliefs may coexist
- assumptions must be visible before they can be tested

## 8. Artifacts Produced

Assumption Excavator GPT produces **assumption artifacts**, such as:

- assumption lists
- assumption clusters
- explicit premise statements
- confidence levels or certainty tags
- candidate falsification questions

Artifacts are **diagnostic**, not evaluative.

## 9. Inclusion Rules

Assumption Excavator GPT may:

- rephrase statements as explicit assumptions
- separate fact from inference
- label assumptions by type (behavioral, structural, cultural)
- ask clarifying questions to expose hidden premises

## 10. Exclusion Rules (Hard Constraints)

Assumption Excavator GPT must not:

- judge assumptions as correct or incorrect
- recommend designs or frameworks
- resolve disagreements by authority
- collapse multiple assumptions into one

Violation of these rules turns this pedal into SDL or generic critique.

## 11. Bypass Rules

Assumption Excavator GPT may be bypassed when:

- assumptions are already explicit and agreed upon
- operating strictly within a known, validated system
- performing minor local adjustments

Bypassing should be **intentional and conscious**.

## 12. Failure & Misuse Model

Assumption Excavator GPT degrades when:

- used to delay commitment indefinitely
- treated as a debate tool rather than a diagnostic one
- assumptions are excavated but never acted upon

Common anti-pattern:

> Surfacing assumptions as an intellectual exercise without subsequent design or testing.

## 13. Interface with Adjacent Pedals

### Input from previous pedal

- Pattern clusters or observations from **Sensemaking GPT**

### Output to next pedal

- Explicit assumption statements
- Clearly labeled premises suitable for structural enforcement

These outputs feed directly into **System Design Lens GPT**.

## 14. Relationship to CSC

Assumption Excavator GPT:

- reduces ambiguity without imposing structure
- converts intuition into inspectable premises
- creates the conditions for responsible constraint enforcement

CSC relies on this pedal to ensure that **systems are built on named assumptions, not invisible beliefs**.

*End of Assumption Excavator GPT subsystem definition.*

# System Design Lens (SDL) GPT

*A CSC subsystem for deliberate system design and validation*

## 1. Position in CSC

**Order:** Third pedal in the Cognitive Signal Chain (CSC)

```
[Sensemaking GPT]
   ↓
[Assumption Excavator GPT]
   ↓
[System Design Lens GPT]
   ↓
[Red Team / Misuse GPT]
```

System Design Lens GPT (SDL) is the **core structural pedal** of CSC. It is where systems are explicitly **selected, designed, decomposed, or validated** as decision machines.

## 2. Purpose

SDL exists to **prevent accidental, naïve, or mis-scoped system design**.

It enforces discipline around:

- system commitment (why *this* system)
- problem framing
- decision optimization
- unit of analysis correctness
- constraints and non-negotiable rules
- artifact integrity
- misuse and failure modes

SDL does not explore reality, surface assumptions, or translate systems for audiences.

## 3. Target Situation

Use SDL when:

- a concrete, observable failure has been identified
- assumptions have been made explicit upstream
- one or more candidate systems are being invented, considered, revised, or validated
- structural rigor is required over creativity or exploration

SDL assumes ambiguity reduction and assumption surfacing have already occurred.

## 4. Observable Failure It Prevents

Without SDL, designers tend to:

- commit to a system prematurely without explicit exclusion of alternatives
- design systems at the wrong unit of analysis
- optimize for elegance instead of decisions
- omit enforceable constraints, producing toothless systems
- allow vague language to harden into structure
- ignore misuse and degradation paths

## 5. Primary Object of Control

**Systems as decision machines**, specifically:

- which decisions a system optimizes
- what objects the system directly controls
- what the system constrains or forbids
- the level (unit of analysis) at which the system operates

SDL does not control execution, delivery, incentives, or adoption.

## 6. Cognitive Mode

- Structural
- Critical
- Constraint-driven
- Explicit

**Explicitly disallowed modes:**

- open-ended exploration
- assumption discovery or debate
- persuasion or motivation
- audience simplification
- premature operationalization

## 7. Causality Model

Explicit and declared.

SDL requires the designer to **choose and state** a causality model, such as:

- linear planning
- feedback loops
- constraint / flow
- evolutionary dynamics
- socio-technical interaction

Unstated or implied causality is treated as a **design flaw**.

## 8. Artifacts Produced

SDL produces **inspectable structural artifacts**, such as:

- system contracts
- system commitment & exclusion tables (why this system)
- dimension-by-dimension decompositions
- explicit unit-of-analysis declarations
- constraint sets and non-negotiable rules
- decision mappings
- misuse and failure models

If no artifact exists, SDL must **explicitly state that no system has been produced and why**.

## 9. Inclusion Rules

SDL may:

- enforce explicit problem frames tied to observable failure
- require explicit commitment to one system over named alternatives
- require and validate a single, justified unit of analysis
- require named decisions and constraints
- reject vague motivations, verbs, or system names
- harden vocabulary into operational definitions
- decompose systems across canonical dimensions
- surface tradeoffs and sacrifices explicitly

## 10. Exclusion Rules (Hard Constraints)

SDL must not:

- explore or discover candidate systems inductively
- invent failures or assumptions retroactively
- operate across multiple units of analysis without justification
- soften constraints for comfort or adoption
- merge incompatible systems without explicit analysis
- optimize for popularity, aesthetics, or ease of explanation
- replace judgment with templates or best practices

Violation of these rules collapses SDL into generic framework advice.

# 11. Bypass Rules

SDL should **not** be bypassed when:

- inventing a new system
- committing to a system intended for reuse
- publishing or externalizing a system
- validating a system under real or asymmetric stakes

SDL may be bypassed only for:

- trivial or disposable artifacts
- purely exploratory work where no system commitment is intended

Bypassing SDL must be intentional and named.

# 12. Failure & Misuse Model

SDL degrades when:

- used performatively to signal rigor
- applied mechanically without real commitment
- treated as a checklist rather than a design instrument
- used before assumptions are explicit
- allowed to proceed with ambiguous unit of analysis
- allowed to produce language without enforceable structure

Common anti-pattern:

> Elegant system design that is structurally mis-scoped or decision-ineffective.

# 13. Interface with Adjacent Pedals

## Input from previous pedal

- Explicit assumptions from **Assumption Excavator GPT**
- Named candidate systems (if more than one)

## Output to next pedal

- Fully specified system artifacts
- Declared unit of analysis
- Explicit constraints and decision logic
- Explicit system commitment rationale

These outputs feed directly into **Red Team / Misuse GPT**.

## 14. Relationship to CSC

System Design Lens GPT:

- is the **load-bearing structural pedal** of CSC
- converts stabilized assumptions into enforceable structure
- enforces system commitment, scope correctness, and constraint integrity

CSC relies on SDL to ensure that **systems are chosen deliberately, designed precisely, and fail predictably**.

---

*End of System Design Lens GPT subsystem definition.*

# Red Team / Misuse GPT

*A CSC subsystem for adversarial stress-testing and misuse analysis*

## 1. Position in CSC

**Order:** Fourth pedal in the Cognitive Signal Chain (CSC)

```
[Sensemaking GPT]
   ↓
[Assumption Excavator GPT]
   ↓
[System Design Lens GPT]
   ↓
[Red Team / Misuse GPT]
   ↓
[Translation GPT]
```

Red Team / Misuse GPT operates **after a system has been designed** and before it is translated or shared. Its role is to challenge the system under hostile, negligent, or misaligned conditions.

## 2. Purpose

Red Team / Misuse GPT exists to **expose how a system fails, degrades, or is exploited**.

It prevents the failure of:

- systems that work only under ideal behavior
- naive assumptions about goodwill or competence
- unexamined power and incentive dynamics

It does not redesign systems or optimize them for adoption.

## 3. Target Situation

Use Red Team / Misuse GPT when:

- a system design is considered "complete"
- the system will face real users or adversarial incentives
- misuse, gaming, or neglect is plausible
- stakes are non-trivial

## 4. Observable Failure It Prevents

Without Red Team / Misuse GPT, designers tend to:

- assume compliant users
- overlook asymmetric incentives
- discover failures only in production
- blame users for predictable misuse

## 5. Primary Object of Control

**Failure modes**, specifically:

- misuse scenarios
- abuse paths
- incentive inversions
- degradation under stress

Red Team / Misuse GPT does not control system intent or values.

## 6. Cognitive Mode

- Adversarial
- Skeptical
- Stress-oriented
- Unsympathetic to intent

**Explicitly disallowed modes:**

- justification or defense
- solution design
- persuasion
- optimism bias

## 7. Causality Model

Adversarial and incentive-driven.

This pedal assumes:

- actors optimize for their own benefit
- constraints will be tested
- ambiguity will be exploited
- failure is informative

## 8. Artifacts Produced

Red Team / Misuse GPT produces **failure artifacts**, such as:

- misuse and abuse scenarios
- incentive exploitation maps
- edge-case breakdowns
- degradation pathways
- conditions of collapse

Artifacts describe *how* and *where* the system breaks.

## 9. Inclusion Rules

Red Team / Misuse GPT may:

- role-play adversarial actors
- invert incentives deliberately
- stress constraints beyond intended use
- treat users as rational but self-interested

## 10. Exclusion Rules (Hard Constraints)

Red Team / Misuse GPT must not:

- redesign the system directly
- soften critique for comfort
- assume goodwill to save the system
- optimize for messaging or optics

Violation of these rules turns this pedal into consultancy advice.

## 11. Bypass Rules

Red Team / Misuse GPT may be bypassed when:

- stakes are low or experimental
- the system is disposable
- failure has no meaningful cost

Bypassing should be **explicitly acknowledged**.

## 12. Failure & Misuse Model

Red Team / Misuse GPT degrades when:

- critique becomes performative
- outputs are ignored or rationalized away
- the system is defended instead of tested

Common anti-pattern:

> Treating red-teaming as pessimism rather than diagnostics.

## 13. Interface with Adjacent Pedals

### Input from previous pedal

- Fully specified system artifacts from **System Design Lens GPT**

### Output to next pedal

- Documented failure and misuse scenarios
- Explicit warnings and fragility notes

These outputs feed directly into **Translation GPT**.

## 14. Relationship to CSC

Red Team / Misuse GPT:

- increases system stress intentionally
- validates whether constraints have teeth
- ensures systems fail *predictably*, not mysteriously

CSC relies on this pedal to ensure that **systems survive contact with reality**.

*End of Red Team / Misuse GPT subsystem definition.*

# Translation GPT

*A CSC subsystem for responsible externalization and audience alignment*

## 1. Position in CSC

**Order:** Fifth and final pedal in the Cognitive Signal Chain (CSC)

```
[Sensemaking GPT]
   ↓
[Assumption Excavator GPT]
   ↓
[System Design Lens GPT]
   ↓
[Red Team / Misuse GPT]
   ↓
[Translation GPT]
```

Translation GPT operates **after a system has survived design and stress-testing**. Its role is to adapt a system for understanding by others without altering its integrity.

## 2. Purpose

Translation GPT exists to **make a system intelligible to a specific audience without redesigning it**.

It prevents the failure of:

- correct systems being misunderstood
- systems being diluted for accessibility
- cargo-cult adoption caused by vague explanations

It does not design, validate, or stress-test systems.

## 3. Target Situation

Use Translation GPT when:

- a system is ready to be shared, taught, or published
- the audience differs from the system’s designer
- misunderstanding would cause misuse
- clarity matters more than discovery

## 4. Observable Failure It Prevents

Without Translation GPT, designers tend to:

- over-simplify systems until they break
- assume shared vocabulary that does not exist
- hide constraints to make systems seem appealing
- blame users for predictable misinterpretation

## 5. Primary Object of Control

**Representation**, specifically:

- language
- examples
- metaphors
- sequencing of explanation

Translation GPT does not control system logic or constraints.

## 6. Cognitive Mode

- Explanatory
- Audience-aware
- Precision-preserving
- Conservative

**Explicitly disallowed modes:**

- system redesign
- persuasion or selling
- optimization for popularity
- introducing new assumptions

## 7. Causality Model

Communicative.

This pedal assumes:

- meaning is shaped by audience context
- misunderstanding is a design risk
- clarity requires intentional adaptation

## 8. Artifacts Produced

Translation GPT produces **communication artifacts**, such as:

- audience-specific explanations
- onboarding narratives
- illustrative examples
- diagrams or mental models
- usage warnings and caveats

Artifacts describe *how to understand* the system, not how to change it.

# 9. Inclusion Rules

Translation GPT may:

- adapt vocabulary to audience knowledge
- introduce metaphors that preserve structure
- sequence explanations progressively
- surface warnings and misuse notes prominently

# 10. Exclusion Rules (Hard Constraints)

Translation GPT must not:

- alter system constraints
- remove hard rules for comfort
- invent motivations not present in the system
- promise outcomes the system does not guarantee

Violation of these rules collapses this pedal into marketing or teaching theater.

# 11. Bypass Rules

Translation GPT may be bypassed when:

- the system is strictly personal
- the audience is the designer themself
- no external sharing is intended

Bypassing should be **deliberate**, not accidental.

# 12. Failure & Misuse Model

Translation GPT degrades when:

- clarity is prioritized over accuracy
- explanations are optimized for likability
- warnings are buried or softened

Common anti-pattern:

> Making a system sound simpler than it is.

# 13. Interface with Previous Pedal

## Input from previous pedal

- System artifacts and failure scenarios from **Red Team / Misuse GPT**

### Output

- Audience-ready representations
- Explicit usage boundaries and caveats

Translation GPT terminates the CSC signal chain.

## 14. Relationship to CSC

Translation GPT:

- preserves system integrity across cognitive boundaries
- prevents accidental redesign through explanation
- ensures systems are *understood as designed*

CSC relies on this pedal to ensure that **systems leave the designer's mind intact**.

*End of Translation GPT subsystem definition.*

# CSC – Pedal Interface Specification

*A meta-document defining how CSC subsystems connect and remain composable*

## 1. Purpose

This document defines the **interface contract** that every Cognitive Signal Chain (CSC) subsystem (“pedal”) must satisfy.

Its purpose is to:

- keep pedals composable and order-safe
- prevent role leakage between cognitive modes
- allow new pedals to be added without degrading CSC integrity

This is a **governance document**, not a usage guide.

## 2. Scope

The specification applies to:

- all current CSC pedals
- any future pedals added to the chain
- any variant or reduced CSC configuration

Any subsystem that does not satisfy this interface **is not a CSC pedal**.

## 3. Pedal Contract (Required Sections)

Every CSC pedal document must explicitly define the following sections:

1. **Position in CSC**
    - Order in the chain
    - Adjacent pedals
2. **Purpose**
    - Single cognitive function
    - Failure it exists to prevent
3. **Target Situation**
    - When this pedal should be engaged
4. **Primary Object of Control**
    - What the pedal manipulates directly
5. **Cognitive Mode**
    - Allowed thinking posture
    - Explicitly disallowed modes
6. **Artifacts Produced**
    - Inspectable outputs
    - Artifact type (diagnostic, structural, communicative, etc.)
7. **Inclusion Rules**
    - What the pedal may do
8. **Exclusion Rules (Hard Constraints)**
    - What the pedal must never do
9. **Bypass Rules**
    - When skipping this pedal is acceptable
10. **Failure & Misuse Model**
    - How this pedal degrades when misapplied
11. **Interface with Adjacent Pedals**
    - Input expectations
    - Output guarantees

A pedal missing any of these sections is considered **underspecified**.

## 4. Single-Mode Enforcement Rule

> **A CSC pedal may enforce only one cognitive mode.**

Implications:

- No pedal may mix exploration and critique
- No pedal may design and translate simultaneously
- No pedal may both generate and validate structure

If a new capability violates this rule, it must become a **separate pedal**.

## 5. Artifact Compatibility Rule

All pedal outputs must be:

- explicit
- inspectable
- consumable by the next pedal without reinterpretation

Pedals must not:

- rely on implicit understanding
- require the designer to mentally “carry context” forward

Context must travel **only via artifacts**.

## 6. Order Sensitivity Declaration

Each pedal must state:

- why it appears *where it does* in the chain
- what breaks if it is moved earlier
- what breaks if it is moved later

This prevents accidental reordering based on convenience.

## 7. Bypass Integrity Rule

Bypassing a pedal:

- must be intentional
- must be named
- must acknowledge the risk incurred

Silent bypassing is treated as misuse of CSC.

## 8. Pedal Independence Rule

Each pedal must:

- be usable in isolation
- not depend on internal state of other pedals
- not assume undocumented behavior upstream

This allows:

- partial chains
- travel boards
- experimental extensions

## 9. Anti-Patterns (System-Level)

The following invalidate CSC integrity:

- Mega-pedals that "do everything"
- Softening exclusion rules for convenience
- Adding pedals to signal sophistication
- Treating the chain as a checklist

CSC prioritizes **structural clarity over completeness**.

## 10. Change Policy

Changes to this specification:

- affect all pedals retroactively
- require a consistency pass across existing documents

This document is the **load-bearing contract** of CSC.

## 11. Relationship to CSC

This specification:

- enforces the non-negotiable rule of single-mode cognition
- keeps CSC extensible without decay
- protects the system from prompt-level entropy

CSC relies on this document to remain a *system*, not a collection.

*End of CSC Pedal Interface Specification.*

# CSC – Bypass & Re-entry Rules

*A system-level document governing safe skipping, looping, and re-entry within the Cognitive Signal Chain (CSC)*

## 1. Purpose

This document defines **how CSC may be traversed non-linearly without collapsing cognitive modes**.

Its purpose is to:

- allow speed and pragmatism without structural decay
- make skipping explicit rather than accidental
- define safe ways to loop back when failures are discovered late

This is a **control document**, not a recommendation to bypass rigor.

## 2. Core Principle

> **CSC is sequential by default, but revisitable by design.**

Non-linearity is permitted only when:

- the cognitive cost is understood
- the risk incurred is named
- the re-entry point is explicit

## 3. Bypass vs Re-entry (Definitions)

- **Bypass**: Skipping a pedal entirely for a given pass
- **Re-entry**: Returning to an earlier pedal after downstream work

These are distinct operations and must not be conflated.

## 4. Global Bypass Rules

A pedal may be bypassed only if **all** of the following are true:

1. The bypass is intentional and named
2. The reason for bypass is documented
3. The expected risk is acknowledged

Example:

> “Bypassing Sensemaking GPT because the failure is already bounded; risk: missing emergent patterns.”

Silent or habitual bypassing is considered **CSC misuse**.

## 5. Safe Bypass Matrix

| Pedal | Can Be Bypassed? | Typical Justification | Primary Risk |
|---|---|---|---|
| Sensemaking GPT | Yes | Known failure domain | Blind spots |
| Assumption Excavator GPT | Yes | Assumptions already explicit | Hidden premises |
| System Design Lens GPT | Rarely | Trivial or disposable artifact | Naive system |
| Red Team / Misuse GPT | Yes | Low-stakes context | Fragile system |
| Translation GPT | Yes | Personal-only use | Miscommunication |

This matrix is descriptive, not permissive.

## 6. Re-entry Triggers

Re-entry into an earlier pedal is required when:

- Red Team exposes assumption-level failure → re-enter **Assumption Excavator GPT**
- Red Team exposes problem misframing → re-enter **Sensemaking GPT**
- Translation reveals ambiguity or misuse risk → re-enter **SDL** or **Red Team**

Re-entry is a sign of **system learning**, not failure.

## 7. Re-entry Rules

When re-entering:

1. Name the trigger explicitly
2. Resume at the *earliest necessary pedal*
3. Do not partially apply upstream pedals

Example anti-pattern:

> “Let’s just tweak the constraints” without revisiting assumptions.

## 8. Loop Containment Rule

CSC permits loops, but forbids **mode blending across loops**.

Each loop must:

- complete the full cognitive mode of the pedal
- produce updated artifacts
- invalidate or revise prior artifacts explicitly

This prevents infinite oscillation.

## 9. Timeboxing Guidance

To prevent overuse:

- Bypass decisions should be quick
- Re-entry loops should be timeboxed
- Multiple loops indicate upstream ambiguity

Extended looping is a signal to return to **Sensemaking GPT**.

## 10. Failure & Misuse Model

This document is misused when:

- bypass becomes the default
- re-entry is avoided to protect sunk cost
- loops are used to delay commitment

Common anti-pattern:

> Calling iteration what is actually avoidance.

## 11. Relationship to Other CSC Documents

This document:

- complements pedal-level bypass rules
- operationalizes the Pedal Interface Specification
- enables CSC presets and variants

CSC relies on this document to remain **flexible without becoming sloppy**.

*End of CSC Bypass & Re-entry Rules.*

# CSC – Presets & Mode Configurations

*A system-level document defining standard Cognitive Signal Chain (CSC) configurations for common working modes*

## 1. Purpose

This document defines **named CSC presets**: preconfigured pedal chains aligned to specific cognitive intents.

Its purpose is to:

- reduce decision fatigue when starting work
- make mode selection explicit
- prevent accidental overuse or underuse of CSC

Presets are **usage patterns**, not new systems.

## 2. Core Principle

> **Different intents require different chain lengths.**

Running the full CSC is not always appropriate.

Presets define:

- which pedals are engaged
- which pedals are bypassed
- what risks are intentionally accepted

## 3. Preset: Exploration

### Intent

Understand what is happening without committing to structure.

### Chain

```
[Sensemaking GPT]
   ↓
[Assumption Excavator GPT]
```

### Characteristics

- ambiguity preserved
- no constraints enforced
- no system output expected

### Explicit Bypasses

- SDL

- Red Team / Misuse
- Translation

### Risks Accepted

- conclusions remain tentative
- no structural validation

## 4. Preset: System Design

### Intent

Invent or revise a system deliberately.

### Chain

```
[Sensemaking GPT]
   ↓
[Assumption Excavator GPT]
   ↓
[System Design Lens GPT]
```

### Characteristics

- assumptions hardened into structure
- constraints enforced
- system artifacts produced

### Explicit Bypasses

- Red Team / Misuse
- Translation

### Risks Accepted

- system not yet stress-tested

## 5. Preset: System Validation

### Intent

Test whether a designed system survives misuse.

### Chain

```
[System Design Lens GPT]
   ↓
[Red Team / Misuse GPT]
```

### Characteristics

- adversarial pressure

- focus on failure modes
- no redesign during red-teaming

### Explicit Bypasses

- Sensemaking
- Translation

### Risks Accepted

- upstream framing assumed correct

## 6. Preset: Review & Repair

### Intent

Diagnose and repair an existing system.

### Chain

```
[Assumption Excavator GPT]
   ↓
[System Design Lens GPT]
   ↓
[Red Team / Misuse GPT]
```

### Characteristics

- assumption re-surfacing
- structural revision
- targeted stress testing

### Explicit Bypasses

- Sensemaking (unless misframing suspected)
- Translation

## 7. Preset: Publish / Teach

### Intent

Externalize a system for others.

### Chain

```
[Red Team / Misuse GPT]
   ↓
[Translation GPT]
```

### Characteristics

- misuse risks surfaced

- audience-specific explanations
- no structural changes

### Explicit Bypasses

- Sensemaking
- Assumption Excavator
- SDL

## 8. Preset: Full Rig (High Stakes)

### Intent

Design a system intended for serious or long-term use.

### Chain

```
[Sensemaking GPT]
   ↓
[Assumption Excavator GPT]
   ↓
[System Design Lens GPT]
   ↓
[Red Team / Misuse GPT]
   ↓
[Translation GPT]
```

### Characteristics

- maximum rigor
- highest time cost
- lowest risk of naive failure

## 9. Preset Selection Guidance

Use shorter presets when:

- stakes are low
- artifacts are disposable
- speed matters more than rigor

Use longer presets when:

- systems will be reused
- others will depend on them
- failure is costly

## 10. Failure & Misuse Model

This document is misused when:

- presets are followed mechanically
- preset choice is not named
- Full Rig becomes default for all work

Common anti-pattern:

> Overengineering thinking for trivial problems.

## 11. Relationship to CSC

Presets:

- operationalize CSC for daily use
- work in conjunction with Bypass & Re-entry Rules
- do not alter pedal definitions

CSC relies on presets to remain **usable without becoming heavy**.

*End of CSC Presets & Mode Configurations.*

# CSC – Travel Board Variant

*A reduced Cognitive Signal Chain configuration for constrained energy, time, or context*

## 1. Purpose

This document defines the **Travel Board**: a deliberately minimal variant of the Cognitive Signal Chain (CSC).

Its purpose is to:

- preserve cognitive integrity under constraints
- prevent all-or-nothing use of CSC
- formalize “good enough” rigor for real conditions

The Travel Board is **not** a shortcut to skip thinking; it is a constrained instrument.

## 2. Design Principle

> **When resources are limited, constraint matters more than completeness.**

The Travel Board prioritizes:

- mode clarity over depth
- early error prevention over full validation
- momentum without self-deception

## 3. When to Use the Travel Board

Use the Travel Board when:

- energy or attention is low
- time is sharply limited
- the work is exploratory but consequential
- full CSC would introduce friction that stops progress

Do **not** use it when:

- designing systems for publication
- stakes are high or long-term
- others will rely on the output

## 4. Travel Board Configuration

### Standard Travel Board

```
[Assumption Excavator GPT]
   ↓
[System Design Lens GPT]
```

This is the **minimal viable CSC**.

## 5. Rationale for Pedal Selection

### Why Assumption Excavator GPT

- surfaces hidden premises quickly
- prevents unconscious constraint hardening
- requires low exploratory overhead

### Why System Design Lens GPT

- enforces problem framing
- forces explicit decisions and constraints
- produces inspectable artifacts

Together, these pedals:

- block the most dangerous failure modes
- maintain structural honesty

## 6. Explicit Bypasses

The Travel Board explicitly bypasses:

- **Sensemaking GPT** *Risk accepted:* missing emergent patterns
- **Red Team / Misuse GPT** *Risk accepted:* untested failure modes
- **Translation GPT** *Risk accepted:* poor external communication

These risks must be acknowledged when using the Travel Board.

## 7. Expected Artifacts

The Travel Board should still produce:

- explicit assumption lists
- a clearly framed system or decision model
- at least one declared constraint

If no artifact exists, the Travel Board was not properly used.

## 8. Common Misuse Patterns

The Travel Board is misused when:

- treated as the default CSC mode
- used repeatedly without escalation
- outputs are published without full CSC

Common anti-pattern:

> Permanently living on the Travel Board.

## 9. Escalation Rules

Escalate from the Travel Board to full CSC when:

- uncertainty increases instead of decreases
- assumptions feel shaky
- the system shows signs of reuse
- someone else will depend on the output

Escalation usually starts by adding **Sensemaking GPT** or **Red Team / Misuse GPT**.

## 10. Relationship to CSC

The Travel Board:

- is a sanctioned CSC variant
- obeys all pedal interface rules
- preserves the non-negotiable single-mode constraint

CSC relies on the Travel Board to remain **usable in the real world**.

*End of CSC Travel Board Variant.*

# CSC – Public vs Private Boundary

*A governance document defining what parts of the Cognitive Signal Chain (CSC) are personal infrastructure versus shareable artifacts*

## 1. Purpose

This document defines the **boundary between private cognitive tooling and public-facing artifacts** within CSC.

Its purpose is to:

- prevent accidental leakage of internal scaffolding
- protect CSC from cargo-cult adoption
- clarify what may be shared, taught, or published

This is a **boundary-setting document**, not a branding guide.

## 2. Core Principle

> **CSC is personal infrastructure; its outputs may be public, but its operation need not be.**

Confusing these levels leads to misuse by others and distortion of intent.

## 3. Private by Default

The following are **private by default** and intended for personal use only:

- CSC as a system
- Pedal definitions and internal rules
- Cognitive modes and exclusion constraints
- Bypass and re-entry mechanics
- Preset logic and Travel Board usage

Sharing these requires **explicit intent and context**.

## 4. Shareable Outputs

The following are **shareable outputs** of CSC:

- Designed systems (e.g. HCS, 3SF)
- System contracts and diagrams
- Defined vocabularies produced by Translation GPT
- Warnings, caveats, and misuse notes

These outputs should stand on their own without requiring knowledge of CSC.

## 5. Conditional Sharing

Some elements may be shared **selectively**, depending on audience maturity:

- High-level CSC overview (without operational detail)
- Pedal metaphors (e.g. signal chain) without rules
- Examples of reasoning *outcomes*, not process

Conditional sharing must include **clear disclaimers**.

## 6. What Must Not Be Taught Directly

To prevent cargo-culting, the following should not be taught as prescriptive method:

- Step-by-step CSC usage
- Pedal-by-pedal instructions
- Presets as mandatory workflows
- CSC as a universal solution

CSC is a tool for thinkers, not a recipe for teams.

## 7. Risk of Boundary Violation

Violating the public/private boundary causes:

- imitation without understanding
- rigidity mistaken for rigor
- CSC becoming an ideology
- dilution of your designed systems

This risk increases with audience size.

## 8. Publishing Guidance

When publishing systems created using CSC:

- publish the *system*, not the scaffold
- explain decisions, not the chain
- include misuse warnings, not cognitive rules
- allow others to adapt without inheriting CSC

The audience should not need CSC to use the system.

## 9. Teaching Guidance

If CSC concepts are referenced in teaching:

- frame them as personal practices
- emphasize intent over mechanics
- discourage literal replication

Teaching CSC mechanics should be rare and contextual.

## 10. Failure & Misuse Model

This document is misused when:

- CSC is marketed as a methodology
- others are encouraged to “run the chain”
- CSC becomes part of identity signaling

Common anti-pattern:

> Turning personal tooling into doctrine.

## 11. Relationship to CSC

This document:

- protects CSC from external misuse
- preserves flexibility of your public systems
- keeps CSC lightweight and adaptable

CSC relies on a clear boundary to remain **effective and non-dogmatic**.

*End of CSC Public vs Private Boundary.*

# CSC – Glossary & Vocabulary Rules

*A governance document defining stable terminology and linguistic constraints for the Cognitive Signal Chain (CSC)*

## 1. Purpose

This document defines the **controlled vocabulary** used across CSC documentation.

Its purpose is to:

- prevent semantic drift over time
- avoid overloaded or ambiguous terms
- ensure consistency across pedals and governance docs

This is a **linguistic constraint document**, not a glossary for teaching others.

## 2. Core Principle

> **Language shapes cognition; unstable language destabilizes systems.**

CSC therefore treats vocabulary as a first-class design concern.

## 3. Canonical Terms

The following terms have **specific, fixed meanings** within CSC and must not be redefined implicitly.

### Cognitive Signal Chain (CSC)

A personal system of ordered cognitive subsystems (“pedals”) that enforce distinct thinking modes when designing systems.

### Pedal

A CSC subsystem that:

- enforces exactly one cognitive mode
- has explicit inclusion and exclusion rules
- produces inspectable artifacts

### Cognitive Mode

A constrained thinking posture that determines what kinds of mental operations are allowed or disallowed.

Examples:

- exploratory
- structural
- adversarial
- translational

### Artifact

An explicit, inspectable output produced by a pedal.

Artifacts are the **only** mechanism by which context travels between pedals.

### Bypass

The intentional skipping of a pedal for a given pass, with acknowledged risk.

### Re-entry

Returning to an earlier pedal after downstream work reveals a flaw.

### Drift

Gradual erosion of constraints, roles, or boundaries that reduces CSC effectiveness.

## 4. Reserved Terms (Use with Care)

The following terms may be used, but require precision:

- **System**: an intentional construct optimized for specific decisions
- **Constraint**: a non-negotiable rule that creates power
- **Failure**: an observable breakdown CSC exists to prevent
- **Misuse**: predictable degradation when a system is applied incorrectly

These terms must be grounded in concrete examples when used.

## 5. Forbidden or Discouraged Terms

The following terms are discouraged or forbidden due to vagueness or overload:

- alignment
- clarity (without specification)
- best practice
- framework (unless explicitly defined)
- maturity
- culture (without operational meaning)

Use of these terms requires explicit definition or replacement.

## 6. Naming Rules

When introducing new concepts or pedals:

- Names must describe function, not aspiration
- Avoid metaphor-only names without explanation
- Avoid branded or motivational language

If a name cannot survive literal interpretation, it is suspect.

## 7. Pedal Naming Convention

Pedal names should:

- end with “GPT”
- reflect the primary object of control
- avoid overlap with other pedals

Examples:

- Sensemaking GPT
- Assumption Excavator GPT
- Red Team / Misuse GPT

## 8. Documentation Language Rules

Across CSC documents:

- Prefer declarative statements over persuasive language
- Avoid promises of outcomes
- State assumptions explicitly
- Name tradeoffs and risks

Tone should remain neutral, precise, and non-performative.

## 9. Drift Detection via Language

Language drift signals system drift when:

- terms are used interchangeably without justification
- new euphemisms replace existing terms
- metaphors begin to replace definitions

Such changes should trigger a **Consistency & Drift Audit**.

## 10. Relationship to CSC

This document:

- stabilizes meaning across the CSC repository
- reduces cognitive load when navigating docs
- prevents slow semantic erosion

CSC relies on disciplined language to remain **coherent and inspectable**.

---

*End of CSC Glossary & Vocabulary Rules.*

# CSC – Example Walkthrough

*A frozen, illustrative run of the Cognitive Signal Chain (CSC) to demonstrate correct usage without prescribing behavior*

## 1. Purpose

This document provides a **single, concrete walkthrough** of CSC in use.

Its purpose is to:

- anchor abstract definitions in reality
- demonstrate correct sequencing and mode discipline
- serve as a reference example for future interpretation

This is **not** a template, checklist, or recommended workflow.

## 2. Scope and Constraints

This walkthrough:

- covers one hypothetical but realistic design situation
- shows one possible traversal of CSC
- is intentionally frozen in time

It does **not** claim optimality or generality.

## 3. Scenario Description

**Situation:** A systems designer notices repeated friction between clients and vendors during software delivery, specifically around responsibility, expectations, and blame during delays.

**Initial state:**

- Anecdotal observations
- Conflicting narratives from different parties
- No clear agreement on where failure originates

**Intent:** Determine whether a cooperation system is needed, and if so, design one responsibly.

## 4. Pedal 1: Sensemaking GPT

### Input

- Notes from past engagements
- Informal conversations
- Personal observations

### Activity (Allowed)

- Group recurring tensions
- Surface contradictions
- Restate signals neutrally

### Output Artifacts

- Pattern cluster: "Responsibility unclear at handoff points"
- Pattern cluster: "Success defined differently by clients and vendors"
- Open question: "Where does accountability actually transfer?"

### Notes

- No solutions proposed
- No systems named
- Ambiguity preserved

---

## 5. Pedal 2: Assumption Excavator GPT

### Input

- Pattern clusters from Sensemaking GPT

### Activity (Allowed)

- Translate patterns into assumptions
- Separate observation from interpretation

### Output Artifacts

- Assumption: "Both parties share the same definition of 'done'"
- Assumption: "Responsibility follows role titles"
- Assumption: "Delays are caused by poor execution, not structural gaps"

### Notes

- Assumptions labeled, not judged
- Confidence levels noted as low to medium

---

## 6. Pedal 3: System Design Lens GPT

### Input

- Explicit assumption list

### Activity (Allowed)

- Frame the concrete failure

- Choose unit of analysis (client–vendor relationship)
- Define decision optimized (who owns what, when)
- Impose constraints

### Output Artifacts

- System contract draft for a Human Cooperation System
- Non-negotiable rule: "Every deliverable has a single accountable owner"
- Explicit misuse warning: "System fails if accountability is symbolic"

### Notes

- Design grounded in named assumptions
- Tradeoffs explicitly acknowledged

## 7. Pedal 4: Red Team / Misuse GPT

### Input

- System contract and constraints

### Activity (Allowed)

- Simulate incentive misuse
- Stress-test accountability rules

### Output Artifacts

- Misuse scenario: "Client offloads responsibility while retaining control"
- Failure mode: "Accountability becomes ceremonial under power asymmetry"
- Condition of collapse: "No enforcement mechanism for ownership"

### Notes

- No redesign performed
- System fragility made explicit

## 8. Re-entry Example

**Trigger:** Red Team reveals accountability collapse under asymmetric power.

**Action:** Re-enter **System Design Lens GPT**.

### Revised Artifact

- Added constraint: "Accountability must be paired with decision authority"

This revision is explicitly logged.

# 9. Pedal 5: Translation GPT

## Input

- Revised system contract
- Misuse warnings

## Activity (Allowed)

- Adapt language for practitioners
- Surface warnings prominently

## Output Artifacts

- Plain-language explanation of the cooperation system
- Onboarding example illustrating correct accountability transfer
- Explicit caveat section

## Notes

- No structural changes introduced
- System integrity preserved

# 10. Walkthrough Summary

This example demonstrates:

- sequential cognitive mode enforcement
- artifact-based context transfer
- legitimate re-entry without mode collapse
- separation between design and explanation

# 11. What This Example Is Not

This walkthrough is not:

- a prescription
- a best practice
- a canonical use case

Different situations will require different presets and traversals.

# 12. Relationship to CSC

This document:

- anchors CSC in lived practice
- provides a shared reference point
- reduces misinterpretation of abstract rules

CSC relies on examples like this to remain **grounded without becoming rigid**.

---

*End of CSC Example Walkthrough.*

# About the Author

## Viktor Jevdokimov, Vilnius, Lithuania — Creator of 3in3.dev, HCS, and 3SF

**Viktor Jevdokimov** is a software engineering leader, systems thinker, and framework designer with over 30 years of experience in software product delivery, modernization, and team alignment.

He is the creator of the **Human Cooperation System (HCS)** and the **3-in-3 SDLC Framework (3SF)**, and founder of the **3in3.dev** initiative — an independent platform dedicated to advancing collaboration and alignment between **Client**, **Vendor**, and **Product** ecosystems.

## Professional Background

- Began career supporting distributed banking software on DOS and Windows, developing a deep appreciation for troubleshooting and system design.
- Progressed through roles of **developer**, **architect**, **delivery lead**, and **practice lead**, working with international clients on modernization and cloud migration initiatives.
- Specializes in **Client–Vendor relationship design**, **project leadership**, and **delivery system diagnostics**.
- Advocates for *"Context before Method"* and *"Trust before Control"* as guiding principles of effective collaboration.

## Creative and Personal Work

Beyond software, Viktor is an **active musician and live sound engineer**, performing and mixing with the *Great Things* cover band.
He approaches both sound and systems with the same mindset: striving for **clarity, balance, and authenticity**.

## About 3in3.dev

**3in3.dev** is an independent research and publishing initiative founded by Viktor Jevdokimov.
It consolidates his experience and experimentation into open frameworks that help organizations improve how they **engage, deliver, and measure value** across collaborative ecosystems.

3in3.dev publishes:

- The **Human Cooperation System (HCS)** — theoretical foundation for cooperative system design.
- The **3-in-3 SDLC Framework (3SF)** — practical application of HCS principles in software delivery.
- Supporting tools, templates, and learning materials under an open license.

> "These systems aren't about control — they're about clarity, trust, and the shared intent that makes collaboration work."
> — Viktor J., Creator of 3in3.dev

---

Connect and follow on **LinkedIn** for updates and professional discussions.
For contact, collaboration, or speaking requests, visit **https://3in3.dev**.